CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

IT347: SOFTWARE ENGINEERING

Dec. 2019 – May 2020

UNIT 1

Introduction to Software and Software Engineering

DEPSTAR

Content

- The Evolution Role of Software
- Software: A crises on the Horizon and Software Myths
- Software Engineering: A Layered Methodology
- Software Process Models
- Component Based development, Process, Product & Process

IS SOFTWARE DEAD ?

If it was dead, you would not be reading Software Engineering

Defining Software

- Software is
  1. Instructions(Computer programs) that when execute provide desired features, functions and performance)
  2. Data structures that enable the programs to adequately manipulate the information
  3. Descriptive information in both hardcopy and virtual forms that describe the operation and use of the programs

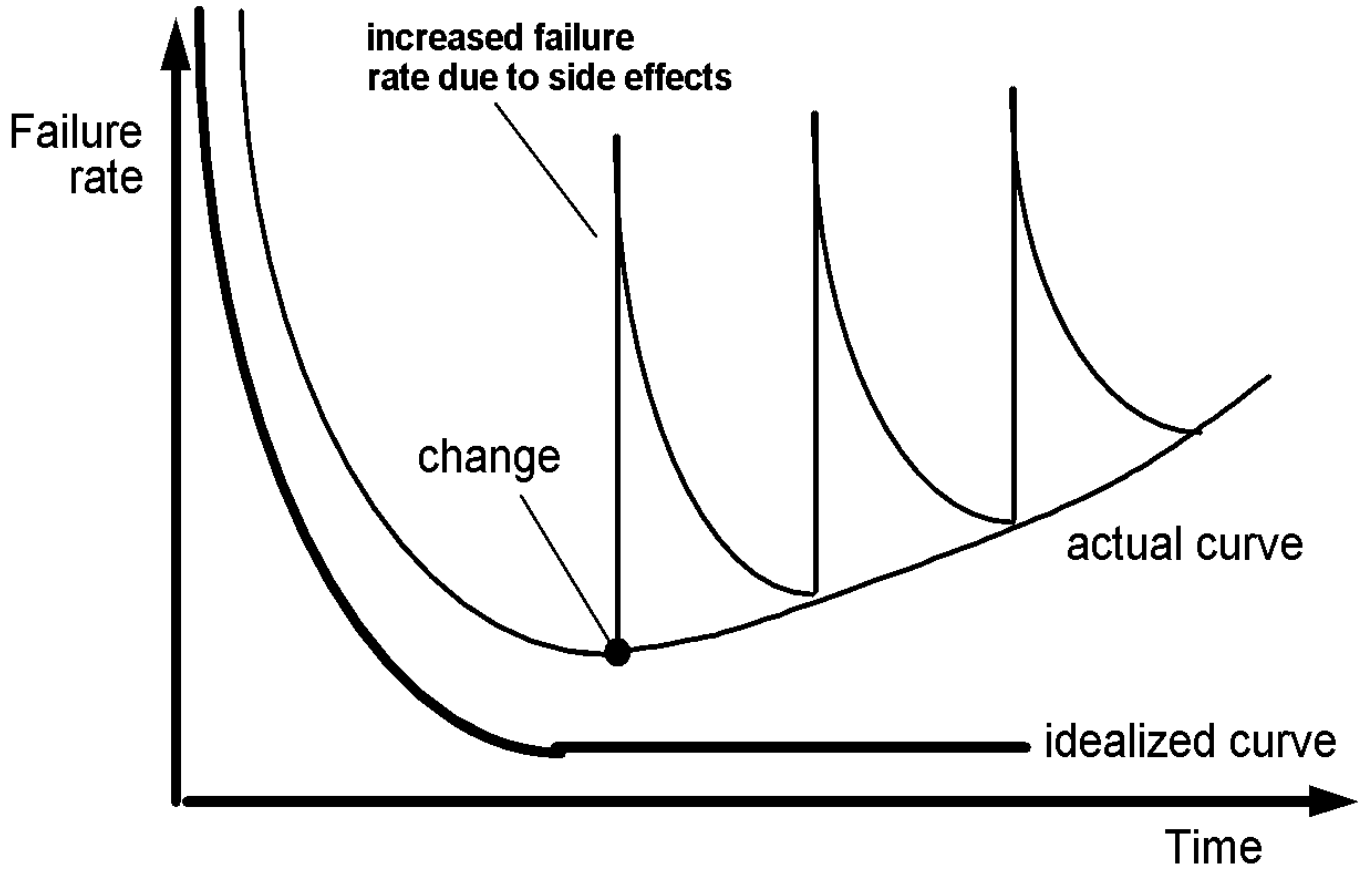
Activity

- Divide in a group of 4
- Give team numbers
- Make a group leader
- All odd teams – topic is hardware
- All even teams – topic is software
- Hardware – design a chair (things required, design, implement, test)
- Software – design any software (things required, design, implement, test)
- Time = 15min

Characteristics of Software

- Software is developed or engineered, it is not manufactured in the classical sense.
- Software doesn't "wear out."
  - As the software is not susceptible to the environmental maladies that cause hardware to wear out
  - Software will undergo changes and as changes are made, errors will be introduced, causing failure rate curve to spike as shown in next slide
- Although the industry is moving toward component-based construction, most software continues to be custom-built.

Wear vs Deterioration



Software Applications

1. System software
2. Application software
3. Engineering/scientific software
4. Embedded software
5. Product-line software
6. WebApps (Web applications)
7. AI software

New Categories

1. Open world computing—pervasive, distributed computing
2. Ubiquitous computing—wireless networks
3. Netsourcing—the Web as a computing engine
4. Open source—"free" source code open to the computing community (a blessing, but also a potential curse!)
5. Also ...
    1. Data mining
    2. Grid computing
    3. Cognitive machines
    4. Software for nanotechnologies

Legacy Software

- Most of the software's fall in the mentioned category but some programs are older, in some cases much older
- These older programs are referred to as legacy software
- *Why must it change?*
  - software must be adapted to meet the needs of new computing environments or technology.
  - software must be enhanced to implement new business requirements.
  - software must be extended to make it interoperable with other more modern systems or databases.
  - software must be re-architected to make it viable within a network environment.

WebApps

- **Network intensiveness.** A WebApp resides on a network and must serve the needs of a diverse community of clients.
- **Concurrency.** A large number of users may access the WebApp at one time.
- **Unpredictable load.** The number of users of the WebApp may vary by orders of magnitude from day to day.
- **Performance.** If a WebApp user must wait too long (for access, for server-side processing, for client-side formatting and display), he or she may decide to go elsewhere.
- **Availability.** Although expectation of 100 percent availability is unreasonable, users of popular WebApps often demand access on a "24/7/365" basis.

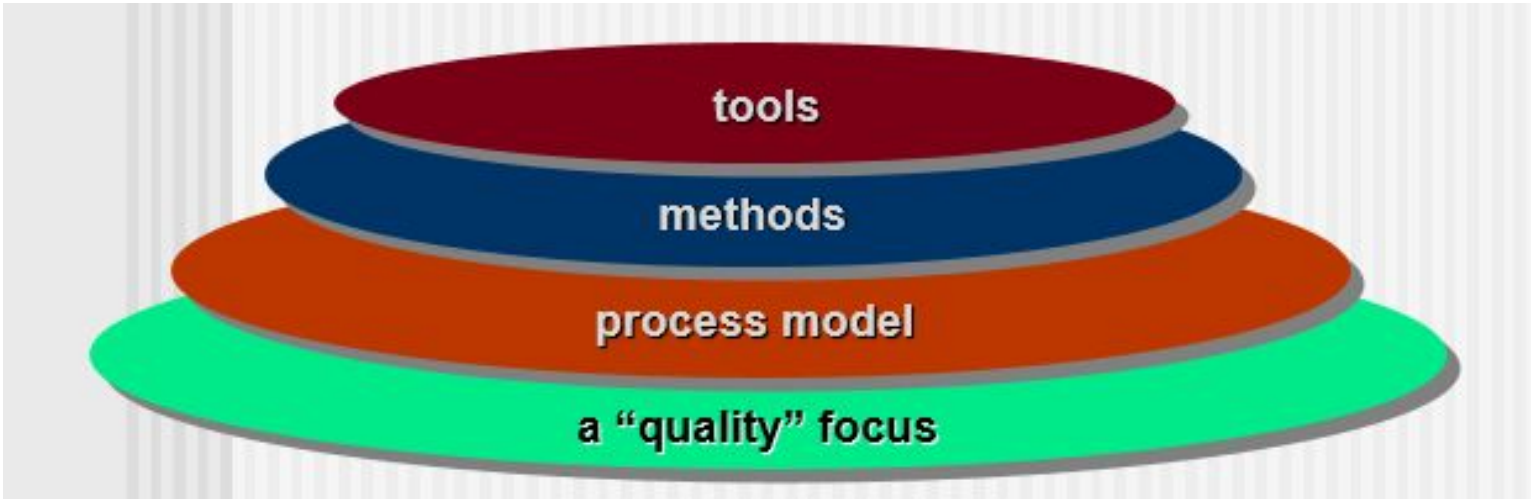
- **Data driven.** The primary function of many WebApps is to use hypermedia to present text, graphics, audio, and video content to the end-user.
- **Content sensitive.** The quality and aesthetic nature of content remains an important determinant of the quality of a WebApp.
- **Continuous evolution.** Unlike conventional application software that evolves over a series of planned, chronologically-spaced releases, Web applications evolve continuously.
- **Immediacy.** Although *immediacy*—the compelling need to get software to market quickly—is a characteristic of many application domains, WebApps often exhibit a time to market that can be a matter of a few days or weeks.
- **Security.** Because WebApps are available via network access, it is difficult, if not impossible, to limit the population of end-users who may access the application.
- **Aesthetics.** An undeniable part of the appeal of a WebApp is its look and feel.

Software Engineering

Definition

- The IEEE definition:
  - *Software Engineering:*
    - *(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.*
    - *(2) The study of approaches as in (1).*

A Layered Technology



A Process Framework

- **Process framework**
  - **Framework activities**
    - work tasks
    - work products
    - milestones & deliverables
    - QA checkpoints
  - **Umbrella Activities**

Framework Activities

- Communication
- Planning
- Modeling
  - Analysis of requirements
  - Design
- Construction
  - Code generation
  - Testing
- Deployment

Umbrella Activities

- Software project management
- Formal technical reviews
- Software quality assurance
- Software configuration management
- Work product preparation and production
- Reusability management
- Measurement

Adapting a process model

- the overall flow of activities, actions, and tasks and the interdependencies among them
- the degree to which actions and tasks are defined within each framework activity
- the degree to which work products are identified and required
- the manner which quality assurance activities are applied
- the manner in which project tracking and control activities are applied
- the overall degree of detail and rigor with which the process is described
- the degree to which the customer and other stakeholders are involved with the project
- the level of autonomy given to the software team
- the degree to which team organization and roles are prescribed

Software Myths

- Software Myths – wrong beliefs about software and the process that is used to build it – can be traced to the earliest days of computing.
- Myths can be:
  - Management Myths
  - Customer Myths
  - Practitioner's Myths

- Management Myths:
  - Already there is a book that is full of standards and procedures for building software. Will not that provide the employees with everything they need to know.
  - In case management is behind schedule, can they add more programmers and catch-up?
  - If decided to outsource the software project to a third party, can management just relax and let that firm build.
- Customer Myths:
  - A general statement of objectives is sufficient to begin writing programs – can the customer fill the details later.
  - Software requirements continuously change, but the change can be easily accommodated because software is flexible.

Contd..

- Practitioner's Myths:
  - Once the program is written and get it to work, the practitioner's work is done.
  - Until the program is got for "running", there is no way to assess the quality
  - The only deliverable work product for a successful project is the working project.
  - SE will make us create voluminous and unnecsary documentation and will invariable slow it down.

Contd…

- Affect managers, customers (and other non-technical stakeholders) and practitioners
- Are believable because they often have elements of truth,

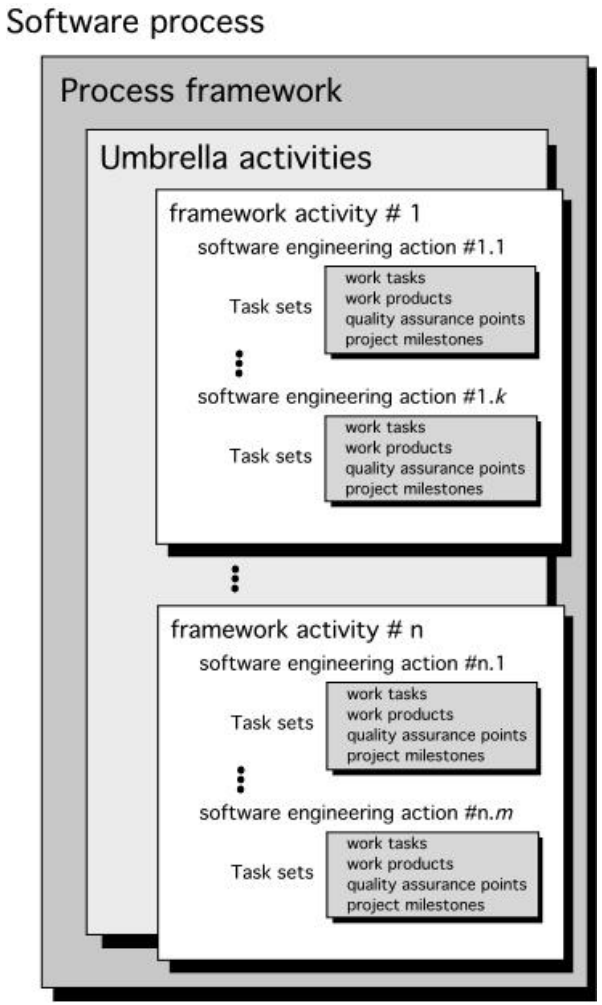
but …

- Invariably lead to bad decisions,

therefore …

- Insist on reality as you navigate your way through software engineering

Process Models



Generic Process Model

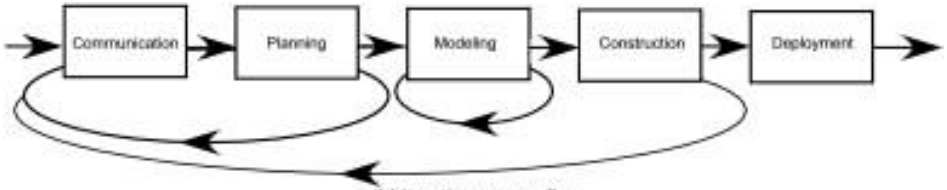
Framework activities of process model

Mainly there are 5 framework activities:

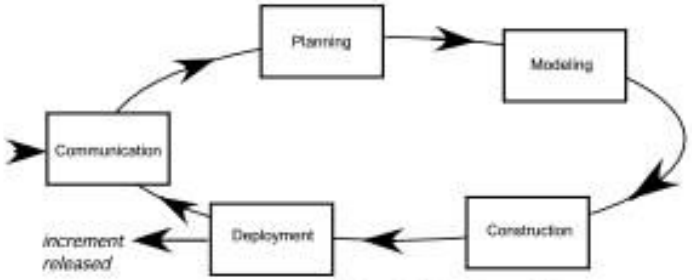
1. Communication
2. Planning
3. Modeling
4. Construction
5. Deployment

Process Flow

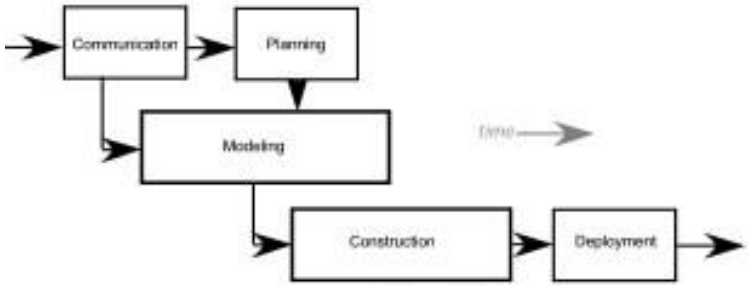
(a) linear process flow



(b) iterative process flow



(c) evolutionary process flow



(d) parallel process flow

Question

- How does a framework activity change as the nature of work changes?

(Hint) : Discuss with respect to a small project and a considerably more complex project with many stake-holders

Answer:

- For small software project requested by 1 person with simple, straightforward requirements, the communication can be:
  - Make contact with stakeholder via telephone
  - Discuss requirements and take notes
  - Organize the notes into a brief written statement of requirements.
  - E-mail to stakeholder for review and approval.

Contd…

- For considerably more complex project with multiple stakeholders:
  - Mindset of every stakeholder may be different.
  - Communication activity might have 6 distinct actions:
    - Inception
    - Elicitation
    - Elaboration
    - Negotiation
    - Specification
    - Validation

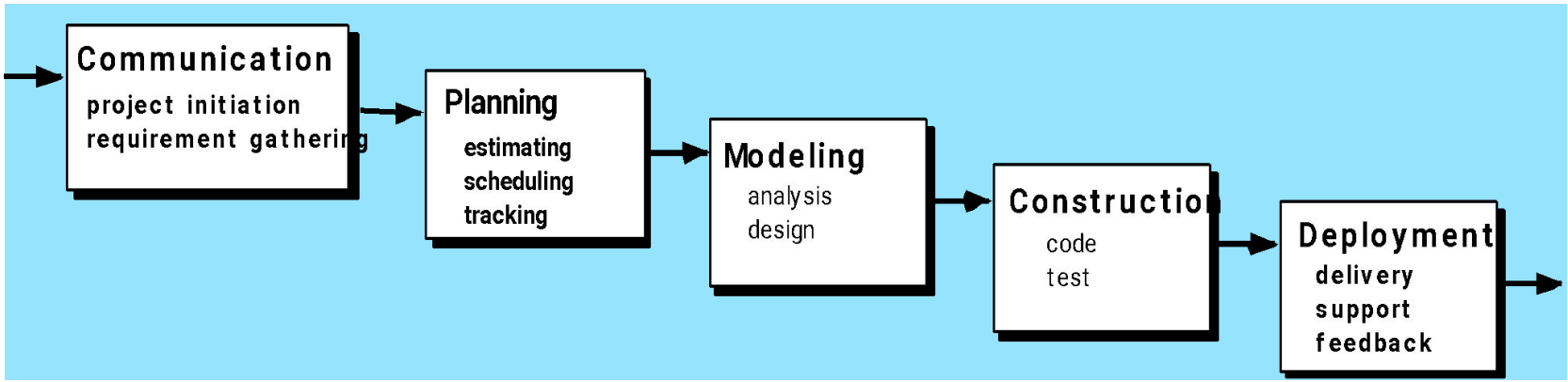
Prescriptive Models

- Prescriptive process models advocate an orderly approach to software engineering

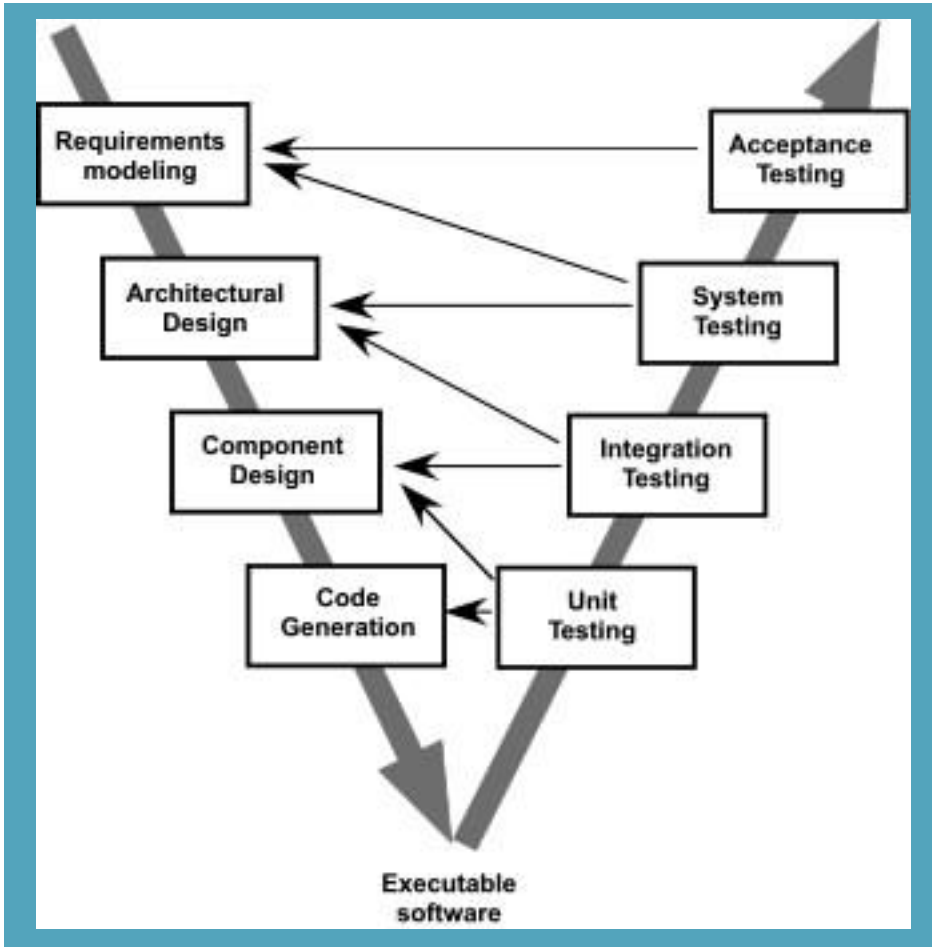
That leads to a few questions …

- If prescriptive process models strive for structure and order, are they inappropriate for a software world that thrives on change?
- Yet, if we reject traditional process models (and the order they imply) and replace them with something less structured, do we make it impossible to achieve coordination and coherence in software work?

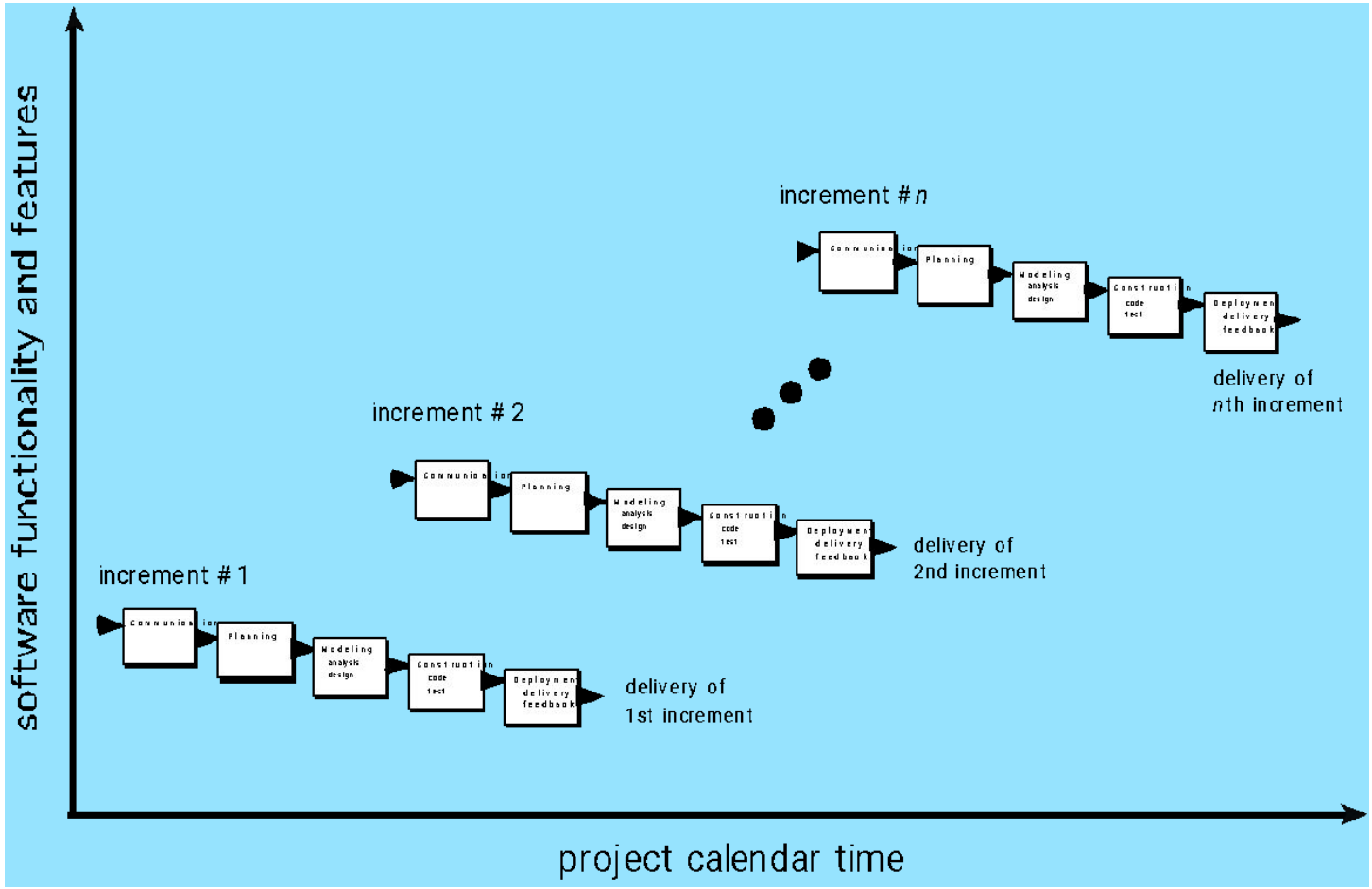
The Waterfall Model



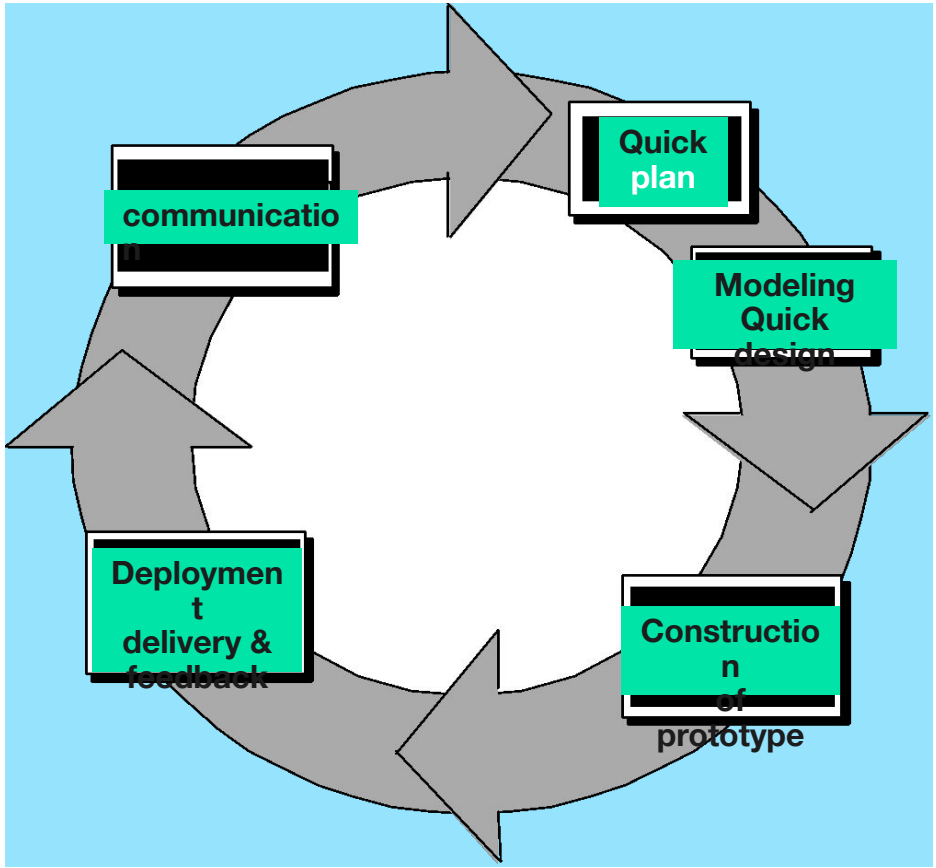
The V-Model


Requirements modeling
Acceptance Testing
Architectural Design
System Testing
Component Design
Integration Testing
Code Generation
Unit Testing
Executable software

The Incremental Model



Evolutionary Models: Prototyping



RAD Model [Rapid Application Development Model]

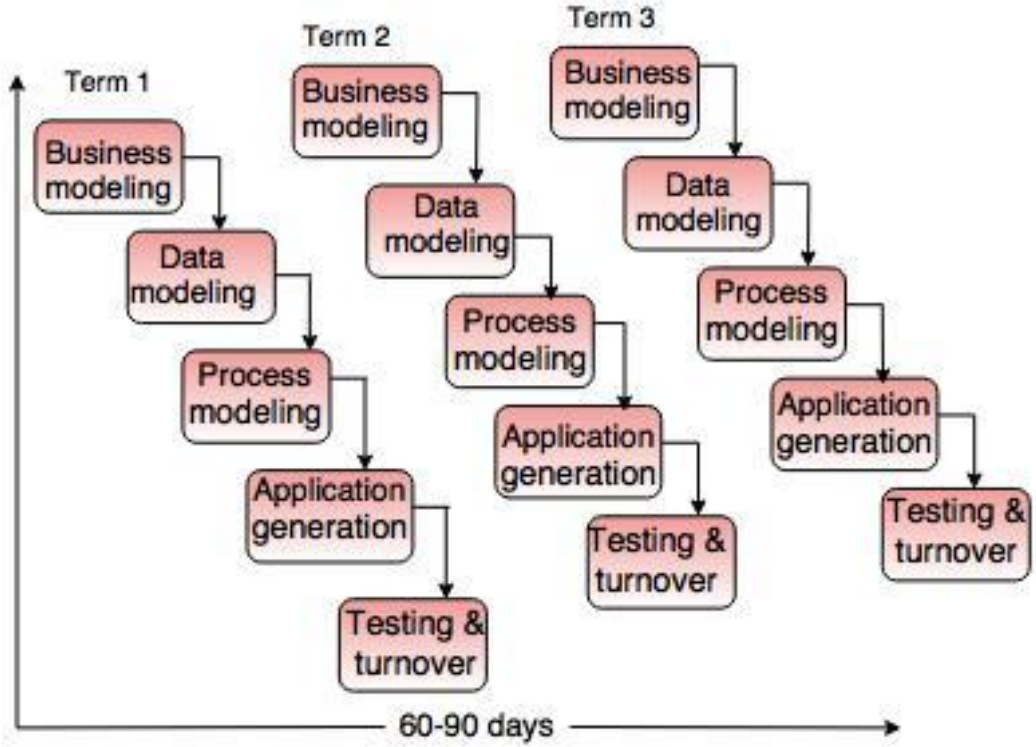
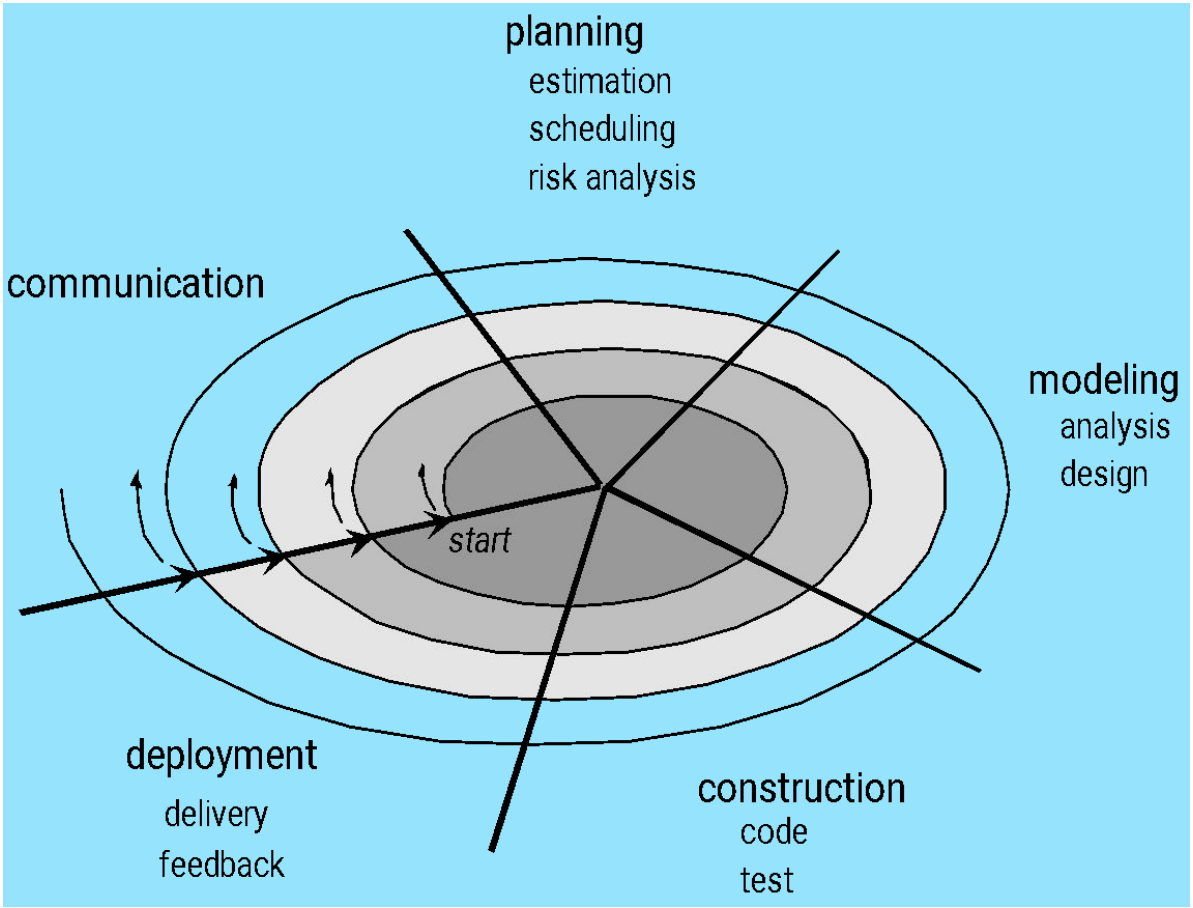
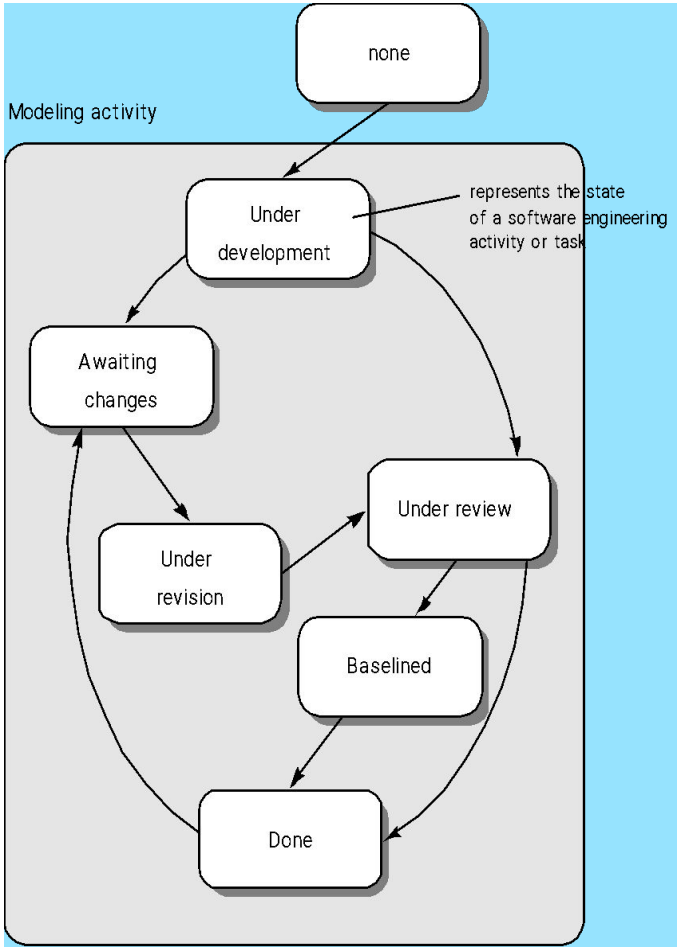


Fig. - RAD Model

Evolutionary Models: The Spiral



Evolutionary Models: Concurrent



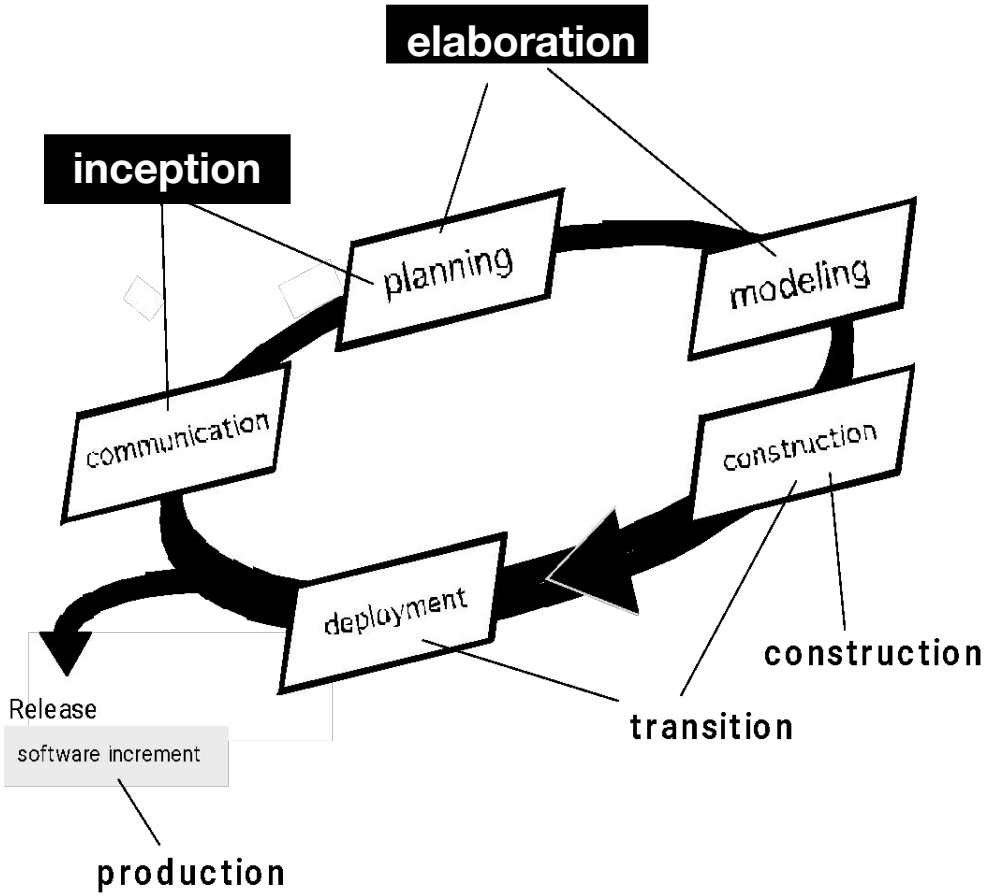
Agile Process Model

AGILE SOFTWARE DEVELOPMENT

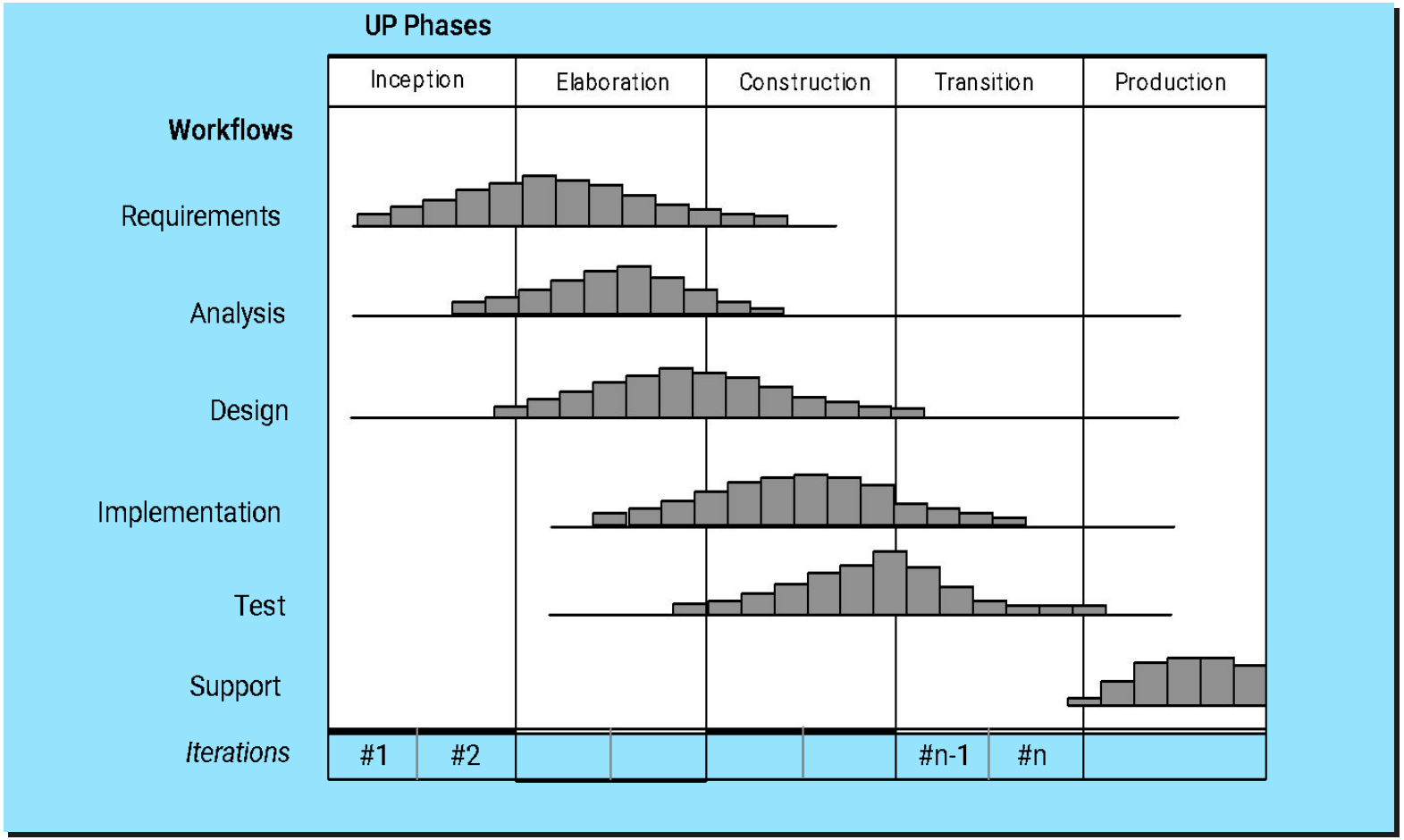
Still Other Process Models

- Component based development—the process to apply when reuse is a development objective
- Formal methods—emphasizes the mathematical specification of requirements
- AOSD—provides a process and methodological approach for defining, specifying, designing, and constructing *aspects*
- Unified Process—a "use-case driven, architecture-centric, iterative and incremental" software process closely aligned with the Unified Modeling Language (UML)

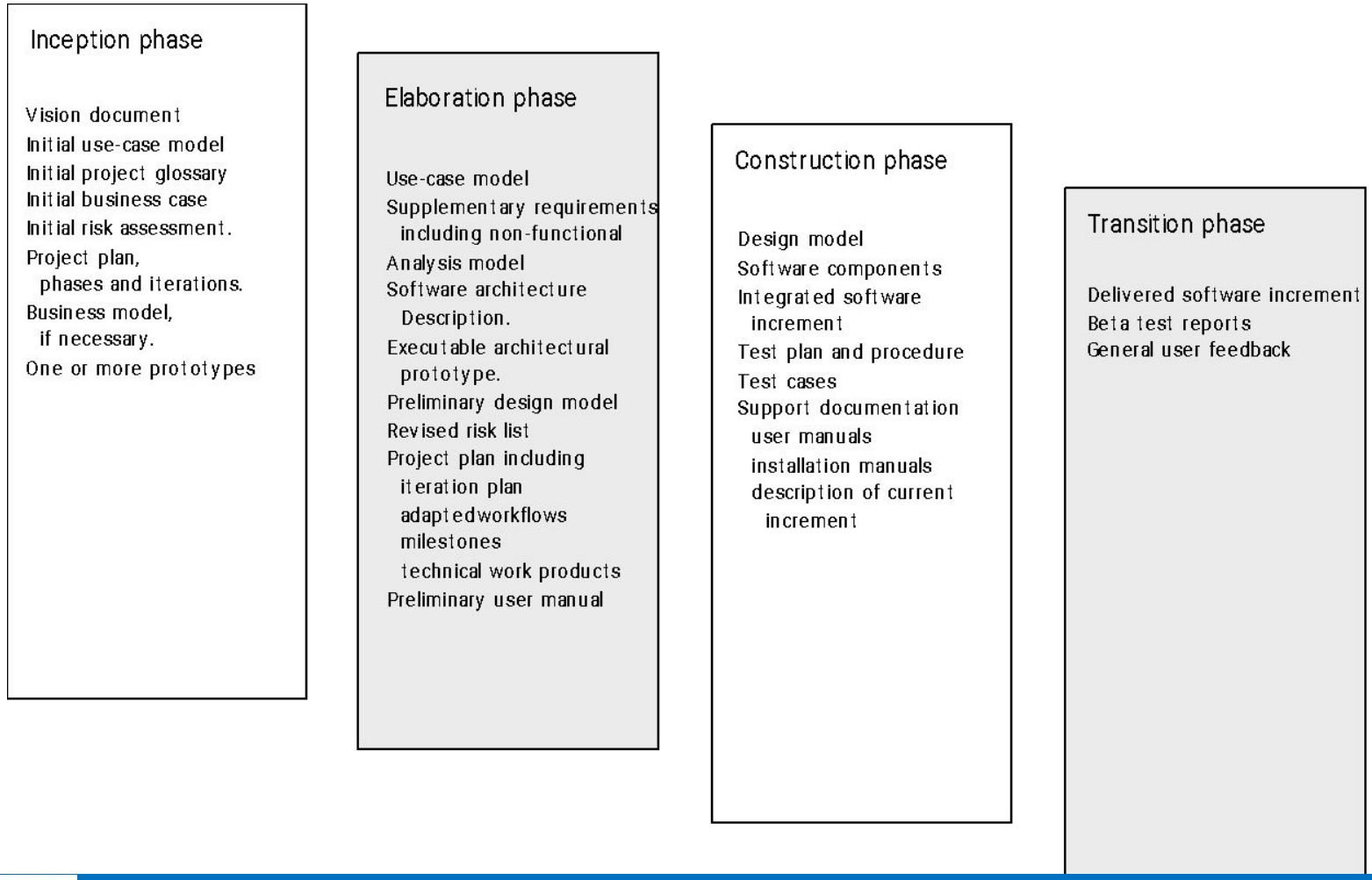
The Unified Process (UP)



UP Phases


UP Phases
Inception
Elaboration
Construction
Transition
Production
Workflows
Requirements
Analysis
Design
Implementation
Test
Support
Iterations
#1
#2
#n-1
#n

UP Work Products



Personal Software Process (PSP)

- **Planning.** This activity isolates requirements and develops both size and resource estimates. In addition, a defect estimate (the number of defects projected for the work) is made. All metrics are recorded on worksheets or templates. Finally, development tasks are identified and a project schedule is created.
- **High-level design.** External specifications for each component to be constructed are developed and a component design is created. Prototypes are built when uncertainty exists. All issues are recorded and tracked.
- **High-level design review.** Formal verification methods (Chapter 21) are applied to uncover errors in the design. Metrics are maintained for all important tasks and work results.
- **Development.** The component level design is refined and reviewed. Code is generated, reviewed, compiled, and tested. Metrics are maintained for all important tasks and work results.
- **Postmortem.** Using the measures and metrics collected (this is a substantial amount of data that should be analyzed statistically), the effectiveness of the process is determined. Measures and metrics should provide guidance for modifying the process to improve its effectiveness.

Team Software Process (TSP)

- Build self-directed teams that plan and track their work, establish goals, and own their processes and plans. These can be pure software teams or integrated product teams (IPT) of three to about 20 engineers.
- Show managers how to coach and motivate their teams and how to help them sustain peak performance.
- Accelerate software process improvement by making CMM Level 5 behavior normal and expected.
  - The Capability Maturity Model (CMM), a measure of the effectiveness of a software process, is discussed in Chapter 30.
- Provide improvement guidance to high-maturity organizations.
- Facilitate university teaching of industrial-grade team skills.


Thank you!